ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laboratorio di

Sicurezza Informatica

OpenSSL

Andrea Melis
Marco Prandini
Dipartimento di Informatica – Scienza e Ingegneria

OpenSSL utility

- OpenSSL è una libreria C che implementa le principali operazioni crittografiche come crittografia simmetrica, crittografia a chiave pubblica, firma digitale, funzioni hash e così via ... OpenSSL implementa ovviamente anche il famoso protocollo Secure Socket Layer (SSL).
- OpenSSL è disponibile per un'ampia varietà di piattaforme. Il codice sorgente può essere scaricato da www.openssl.org.
- In questa esercitazione si vedranno i principali utilizzi di openssl e vedremo come creare e usare una Certification Authority (CA) locale.

OpenSSL Versione e Comandi

- **Visualizziamo la versione di openssl con:**

```
openssl version
OpenSSL 3.0.8 7 Feb 2023
```

- **Se in dubbio usare sempre l'help per vedere i comandi che abbiamo a disposizione:**

```
openssl help
Standard commands
Message Digest commands
Cipher commands
```

OpenSSL Comandi principali

- **ca** creare una certifcation authority
- **dgst** calcolare digest con le funzioni hash
- **enc** de/cifrare utilizzando algoritmi a chiave simmetrica
- **genrsa** generare una coppia di chiavi per l'algoritmo RSA
- **password** generare "hashed password"
- **pkcs7** gestire le informazioni secondo lo standard PKCS # 7
- **pkeyutl** de/cifrare o firmare/verificare con algoritmi asimmetrici
- **rand** generare di stringhe di bit pseudo-casuali.
- **rsa** manipolare chiavi per l'algoritmo RSA
- **verify** verificare certificati (e catene) X509
- **x509** manipolare certificati x509

Algoritmi a chiave simmetrica

- **Quali algoritmi a chiave simmetrica openssl supporta? Vediamolo con:**

```
openssl enc -ciphers
Supported ciphers:
-aes-128-cbc    -aes-128-cfb      -aes-128-cfb1
-aes-128-cfb8   -aes-128-ctr      -aes-128-ecb
...
```

Cifrare un file

- Tra i vari algoritmi di cifratura openssl supporta anche base64
-
- Creiamo un file

```
touch file_base64.txt
echo "Test base64" > file_base64.txt
```

- "Cifriamolo" in base64

```
openssl enc -base64 -in file_base64.txt
U2UgdmVkaSBxdWVzdG8gw6ggcGVyY2jDqCBmYWkgY29waWEv
aW5jb2xsYSBkZWkgY29tYW5kaSBkYWxsZSBzbGlkZSBjb211
IHVuYSBzY2ltbWlhISBzZWUtPiBodHRwczovL3d3dy55b3V0
dWJlLmNvbS93YXRjaD92PTFCaXg0NEMxRXpZCg==
```

- Cosa manca per poterla considerare una cifratura?

Cifrare un file

- **Usiamo quindi un vero algoritmo di cifratura, come AES**

```
openssl enc -aes-256-cbc -md sha512 -pbkdf2 -
iter 100000 -salt -in file_testo.txt -out
cifrato.bin
```

`-aes-256-cbc:` **algoritmo aes 256 CBC, molto sicuro**

`-md sha512:` **algoritmo di hash, della famiglia dello sha2**

`-pbkdf2:` **usa l'algoritmo Password-Based Key Derivation Function 2 per la chiave**

`-iter 100000` **numero di iterazioni necessarie per derivare la chiave. Più è alto il numero più tempo è necessario per fare il brute-force del file, e noi siamo paranoici!**

Decifriamo un file

- Il comando è molto simile è sufficente aggiungere -d e passare in input il file cifrato

```
openssl enc -aes-256-cbc -md sha512 -pbkdf2 -
iter 100000 -salt -d -in cifrato.bin
Inserire la password usata in precedenza
```

- È possibile inoltre inserire la password inline (sconsigliato!)

```
openssl enc -aes-256-cbc -md sha512 -pbkdf2 -
iter 100000 -salt -d -in cifrato.bin -pass
pass:"vostra password precedente"
```

Algoritmi a chiave pubblica

- Generiamo una chiave privata

```
openssl genrsa -out chiave.pem 2048
Generating RSA private key, 2048 bit long
modulus (2 primes)
   ..........+++++
   ..........+++++
   e is 65537 (0x010001)
```

- Grandezza minima (per reputarsi sicura) della chiave è 2048 bit!

Algoritmi a chiave pubblica

- Guardiamo il formato della chiave

```
cat chiave.pem
   Inizia con
      ----BEGIN RSA PRIVATE KEY-----
      Del base64 …
   Finisce con
      ----END RSA PRIVATE KEY-----
```

- Da notare che la chiave che avete generato contiene SIA la chiave privata SIA la chiave pubblica, quest'ultima va però estratta/generata a partire dal file chiave.pem
- Ma prima di tutto analizziamo il file chiave.pem

Algoritmi a chiave pubblica

- Guardiamo i dettagli della chiave con:

```
openssl rsa -in chiave.pem -text -noout
   RSA Private-Key: (2048 bit, 2 primes)
   Modulus:
   publicExponent: 65537 (0x10001)
   PrivateExponent:
   Prime1:
   Prime2:
   Exponent1:
   Exponent2:
   coefficient:
```

- Il -noout ci permette di non visualizzare la chiave in base64

Algoritmi a chiave pubblica

- **Il file creato è ovviamente molto importante e va tenuto al sicuro.**
  Perchè quindi non crittarlo con una chiave simmetrica come fatto in precedenza?

```
openssl rsa -in chiave.pem -aes-256-cbc -out
enc_chiave.pem
   writing RSA key
   Enter PEM pass phrase: Passphrase
   Verifying - Enter PEM pass phrase: Passphrase
```

Algoritmi a chiave pubblica

- Ora che abbiamo creato e messo in sicurazza la chiave privata possiamo estrarre la chiave pubblica.

```
openssl rsa -in chiave.pem -pubout -out
pub_chiave.pem
   writing RSA key
```

- L'intestazione della chiave pubblica sarà invece:

```
cat pub_chiave.pem
   -----BEGIN PUBLIC KEY-----
   chiave in base64
   -----END PUBLIC KEY-----
```

Cifriatura con Chiave pubblica/privata

- **Prima cosa crittiamo un file**

```
openssl rsautl -encrypt -in testo.txt -inkey
chiave.pem -out cifrato_rsa.bin
```

  - È possibile in fase di crittazione aggiungere il parametro -pubin dopo la chiave se si vuole specificare soltanto la chiave pubblica.

```
openssl rsautl -encrypt -in testo.txt -inkey
pub_chiave.pem -pubin -out cifrato_rsa.bin
```

Cifriatura con Chiave pubblica/privata

- **Per decifrare è sufficente cambiare encrypt con decrypt e scambiare l'ordine dei file input e output**

```
openssl rsautl -decrypt -in cifrato_rsa.bin
-inkey chiave.pem -out nuova.txt
```

Ovviamente in questo caso va specificata la chiave privata.

Firma con Chiave pubblica/privata

- Il passaggio successivo consiste nel creare una firma digitale e verificarla.
- Non è molto efficiente firmare un file di grandi dimensioni utilizzando direttamente un algoritmo a chiave pubblica.
- Ecco perché prima calcoliamo il digest delle informazioni da firmare.
  - Nota. Nella realtà le cose sono un più complesse.
  - La sicurezza fornita da questo schema (hashing e quindi firma direttamente utilizzando RSA) non è la stessa (è inferiore in effetti) rispetto alla firma diretta dell'intero documento con l'algoritmo RSA.

Firma con Chiave pubblica/privata

- Per fare la firma utilizziamo l'opzione digest di openssl per creare l'hash

```
openssl dgst -md5 -out digestfile testo.txt
```

-md5 è l'algoritmo di hashing, md5, abbastanza robusto. Ci sono ovviamente altre opzioni come sha o ripemd.

digestfile contiene il risultato dell'hash

Per vedere tutti gli hash algoritm disponibili

```
openssl list --digest-commands
```

Firma con Chiave pubblica/privata

- Possiamo quindi ora firmare il digest

```
openssl rsautl -sign -in digestfile -out
digest_firmato -inkey chiave.pem
```

- Possiamo allo stesso tempo verificare la firma del file in questo modo

```
openssl rsautl -verify -in digest_firmato -out
digest_verifica -inkey chiave.pem
```

- Se tutto è andato a buon fine non ci sono differenza tra i due digest

```
diff digestfile digest_verifica
```

  Non produce risultati

PKI (Public Key Infrastracture)

- **Perchè abbiamo bisogno di una PKI?**
  - **Problema del Man In The Middle**
    - Uno dei principali progressi della crittografia a chiave pubblica è risolvere il problema della distribuzione delle chiavi. La crittografia a chiave segreta presuppone che i partecipanti abbiano già concordato un segreto comune
    - Ma come lo gestiscono in pratica? L'invio della chiave attraverso un canale crittografato sembra la soluzione più naturale e pratica, ma ancora una volta abbiamo bisogno di una chiave segreta comune per farlo.
    - Con la crittografia a chiave pubblica le cose sono molto più semplici: se voglio inviare un messaggio a Bob, devo solo trovare la chiave pubblica di Bob (sulla sua homepage, su una directory di chiave pubblica ...) crittografare il messaggio usando questa chiave e inviarlo il risultato a Bob.

PKI (Public Key Infrastracture)

Cosa può andare storto?

- Tuttavia rimane un grosso problema. Cosa succede se un malintenzionato chiamato Marco mi fa credere che la chiave pubblica che possiede è in realtà quella di Bob?
- Semplicemente invierò un messaggio crittografato utilizzando la chiave pubblica di Marco pensando di comunicare con Bob.
- Bob riceverà il messaggio crittografato, risponderà probabilmente con un altro messaggio crittografato utilizzando la chiave pubblica di Marco.
- Questo attacco si chiama "Man in the middle Attack".
- Quindi abbiamo bisogno di un meccanismo per associare in modo affidabile una chiave pubblica all'identità di una persona (nome, numero di carta d'identità ...).
- Uno di questo meccanismo è implementato in PGP. L'idea è che ognuno costruisca la propria rete di fiducia, disponendo di un elenco di chiavi pubbliche affidabili e condividendo queste chiavi.
- L'altra soluzione è l'uso di una PKI.

  PGP → Web of Trust VS PKI → Root CA trust .. questa è la differenza!

PKI (Public Key Infrastracture)

Come funziona una PKI

- PKI è una soluzione centralizzata al problema della fiducia delle chiavi. L'idea è di avere un'entità fidata (organizzazione, società) che farà il lavoro di certificare che una determinata chiave pubblica appartiene realmente a una determinata persona.
- Questa persona deve essere identificata con il suo nome, indirizzo e altre informazioni utili che possono consentire di sapere chi è questa persona. Una volta terminato questo lavoro, la PKI emette un certificato pubblico per questa persona. Questo certificato contiene tra gli altri:
  - Tutte le informazioni necessarie per identificare questa persona (nome, data di nascita, ...).
  - La chiave pubblica di questa persona.
  - La data di creazione del certificato.
  - La data di revoca del certificato (un certificato è valido per 1 o 3 anni in pratica).
  - La firma digitale di tutte queste informazioni precedenti emessa dalla PKI.

Quindi ora, se voglio inviare un messaggio privato a Bob, posso chiedere il suo certificato. Quando ho ricevuto il certificato, devo controllare la firma della PKI che lo ha emesso e la data di revoca.

PKI (Public Key Infrastracture)

- Prima cosa. Guardiamo il file di configurazione di openssl. Nella nostra VM si trova in: `/etc/ssl/openssl.cnf`

- Cosa guardare in particolare
  - [ca] section
  - Dir dove tutti i file cert e chiavi sono memorizzati
  - Policy
  - Policy match
  - Req
  - I parametri richiesti per il certificato

PKI (Public Key Infrastracture)

- Se non cambiate nulla le impostazioni di default creano una demoCA nella dir `./demoCA.`
- In questa esercitazione non useremo quella directory ma specificheremo noi quali chiavi e certificati usare.
- Per prima cosa dobbiamo creare un certificato per la PKI che conterrà una coppia di chiavi pubbliche / private. La chiave privata verrà utilizzata per firmare i certificati.

```
openssl genrsa -out rootCA.key 2048
openssl req -x509 -new -key rootCA.key -days 3650
-out rootCA.pem
```

La coppia di chiavi sarà in `rootCA.key` e il certificato (che NON contiene la chiave privata, solo quella pubblica) viene salvato in `rootCA.pem`. La chiave privata contenuta in `rootCA.key` è crittografata con una password.

PKI (Public Key Infrastracture)

- Può essere utile esportare il certificato della PKI in formato DER per poterlo caricare nel proprio browser.

```
openssl x509 -in rootCA.pem -outform DER -out
cacert.der
```

- A questo punto abbiamo una rootCA che possiamo usare per firmare i certificati degli utenti!

PKI (Public Key Infrastracture)

- Supponiamo che un utente desideri ottenere un certificato dalla PKI. Per farlo deve creare una richiesta di certificato, che conterrà tutte le informazioni necessarie per il certificato (nome, paese, ... e ovviamente la chiave pubblica dell'utente). Questa richiesta di certificato verrà poi inviata alla PKI.

```
openssl genrsa -out client1.key 2048
# versione minimale:
openssl req -new -key client1.key -out client1.csr

# per essere conformi agli standard Web PKI,
# aggiungere:
-addext "subjectAltName = DNS:www.sitename.domain"
```

  - In `client1.key` ci sono le chiavi dell'utente, in `client1.csr` la richiesta di certificato per la chiave pubblica da firmare

PKI (Public Key Infrastracture)

- Generiamo a questo punto il certificato del client firmato con la chiave della CA

```
openssl x509 -req -days 365 \
-CA rootCA.pem -CAkey rootCA.key \
-CAcreateserial -CAserial serial \
-copy_extensions copy \
-in client1.csr -out client1.pem
   Signature ok
   subject=C = AU, ST = Some-State, O = Internet
   Widgits Pty Ltd
   Getting CA Private Key
```

PKI (Public Key Infrastracture)

- Possiamo a questo punto visualizzare e verificare il certificato con

```
openssl x509 -in client1.pem -text -noout

openssl verify -verbose -CAfile rootCA.pem
client1.pem
```

Test

- Importate il certificato della CA nel browser
- Verificate se è necessario assegnare manualmente il livello di fiducia
- Lato server web apache
  - Come root

```
cd /etc/apache2/sites-enabled
ln -s ../sites-available/default-ssl.conf
vi default-ssl.conf
```

  - Osservate le direttive che identificano i file delle chiavi e dei certificati, modificatele per usare “client1” e la vostra CA, copiando poi i file nelle directory opportune.
  - Avviate il server

```
systemctl start apache2
```